
Contenido

PREPARE TUTORIAL ENVIRONMENT	2
1. – YOU WILL NEED	2
2.- IMPORT PROJECT IN ECLIPSE	3
Steps	3
3. - INSTALLED AND CONFIGURE TIMESTEN DB.....	4
Downloading	4
Configure a DSN(Data Source Name)	4
Info	6
4. - CREATE TOMCAT SERVER IN ECLIPSE	6
TECHNOLOGY INFORMATION	10
GIT/SVN/MERCURIAL	10
The difference between svn (Subversion) and git.	10
Git command line	10
SVN command line	10
JQUERY/ JQUERYUI.....	10
¡IMPORTANT TO KNOW!	10
CODING PRACTICES	10
MAVEN	12
PHASES STARTEGY.....	12
PLUGINS.....	13
JSP.....	14
STRUTS	15
MVC	15
STRUTS PROJECT BASIC CONFIGURATION	15
GLOBAL EXCEPTIONS.....	16
ACTION MAPPINGS	17
STRUTS JSP TAGS.....	22
STRUTS INTEGRATION TESTING	22
INFO.....	23
CLEAN CODE	24

PREPARE TUTORIAL ENVIRONMENT

1. – YOU WILL NEED

GIT installed. <https://git-scm.com/downloads>

Tortoise SVN (Optional) installed. <https://tortoisesvn.net/>

JAVA (JDK) installed.

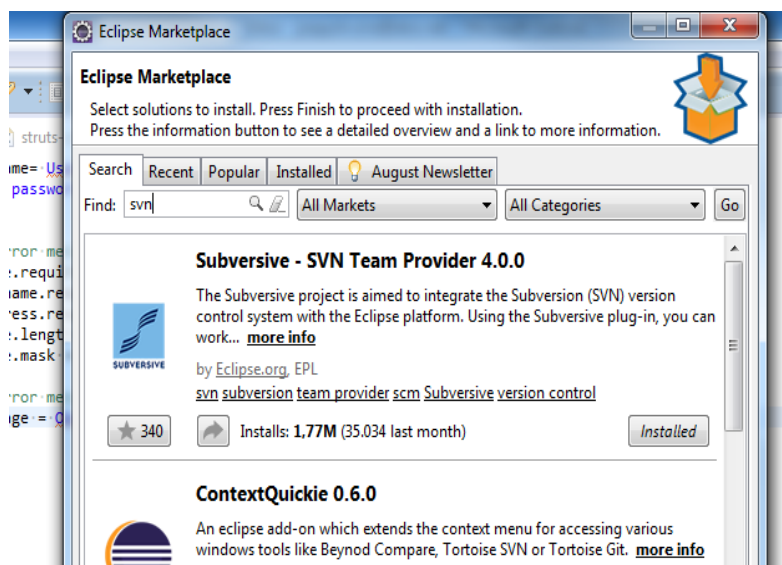
(<http://www.oracle.com/technetwork/java/javase/downloads/jdk8/downloads-2133151.html>)

Maven. Go to <https://maven.apache.org/install.html> for installing steps and go to <http://archive.apache.org/dist/maven/binaries/> for downloading.

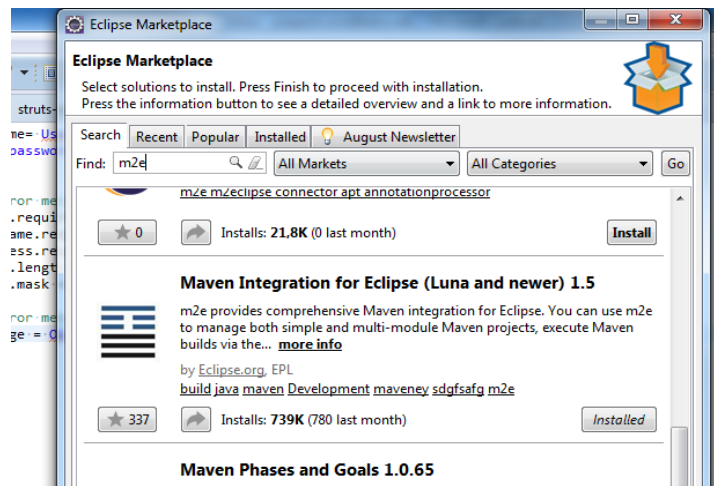
Eclipse installed.

http://www.eclipse.org/downloads/download.php?file=/technology/epp/downloads/release/mars/R/eclipse-je-e-mars-R-win32-x86_64.zip

SVN (SVN Kit) Plugin installed. (**Subversive – SVN Team Provider 4.0.0** from the eclipse marketplace)



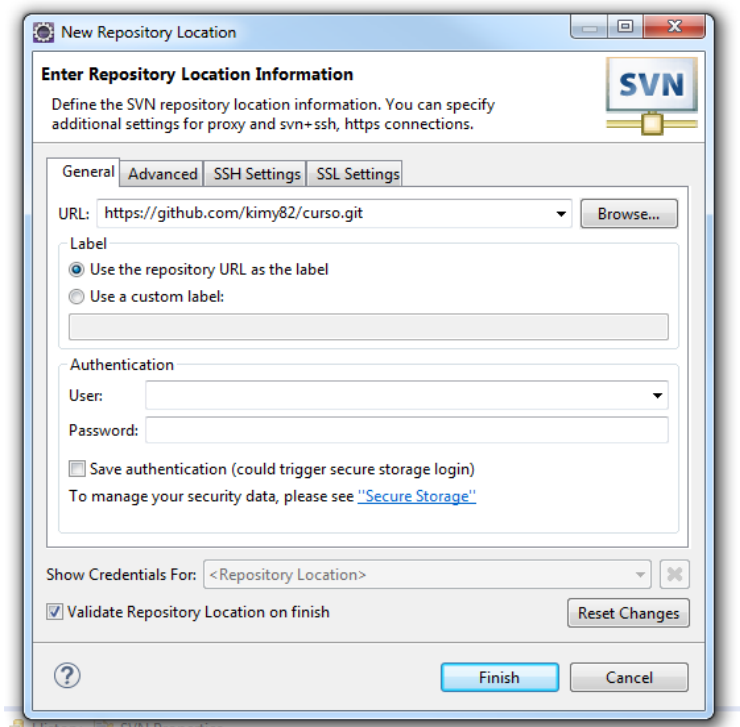
m2e (Maven integration for Eclipse from the eclipse marketplace)



2.- IMPORT PROJECT IN ECLIPSE

Steps

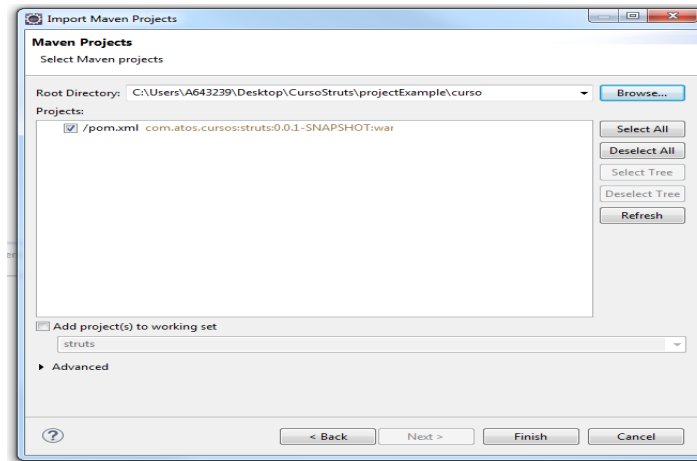
- In eclipse go to “SVN Repository Exploring”.
 - Window -> perspective -> Open perspective ->
- Click right button in the SVN Repositories window and select “new repository location...”. Fill the URL and click finish.



- From the trunk folder right click and select checkout.
- Go to project explorer window. Right click on the project and select configure -> convert to maven project.
- To check that everything is ok. Run a “*mvn install*”. You should see a BUILD SUCCESS.

2st Option (Using GIT)

- From command line type “git clone <projectURL>”. This will download the project.
- Import in Eclipse from cloned project as existing maven project.



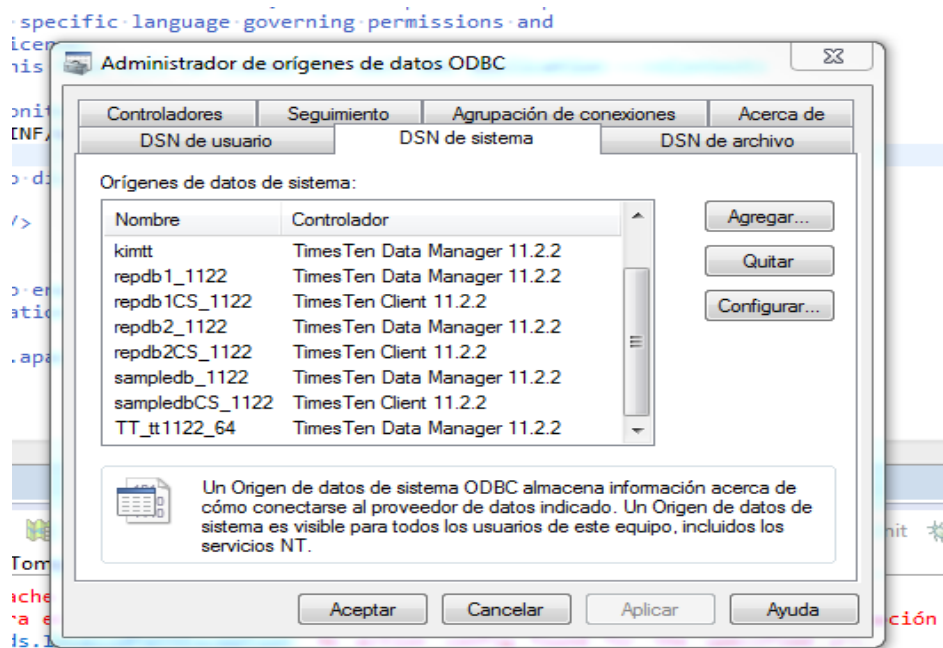
3. - INSTALLED AND CONFIGURE TIMESTEN DB

Downloading

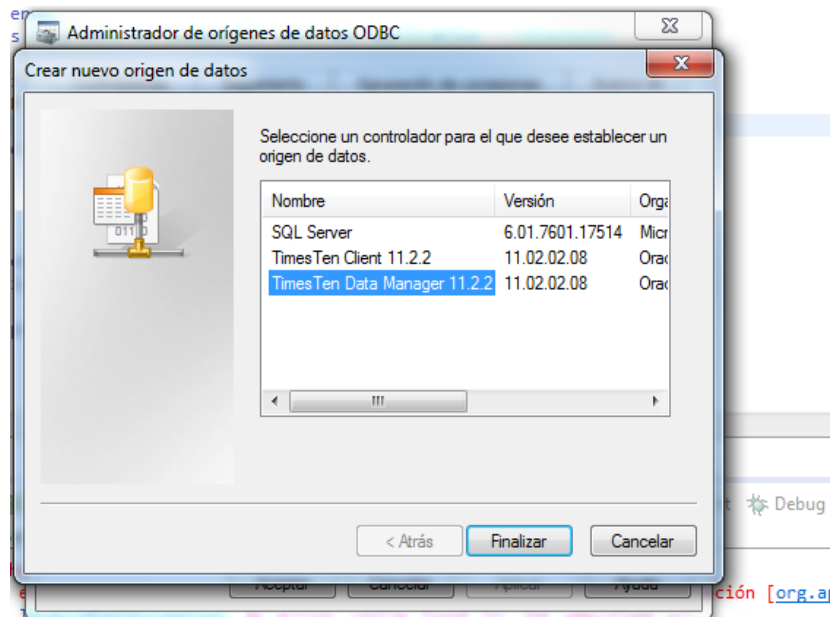
Download times ten from: <http://www.oracle.com/technetwork/database/database-technologies/timesten/downloads/index.html>

Configure a DSN(Data Source Name)

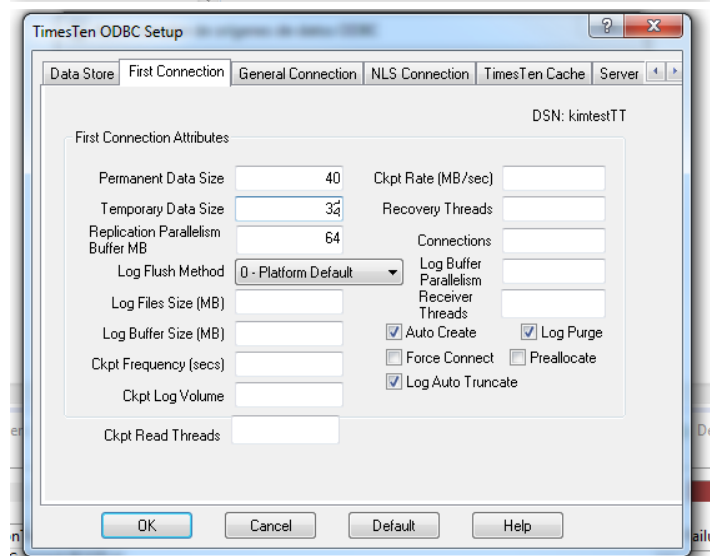
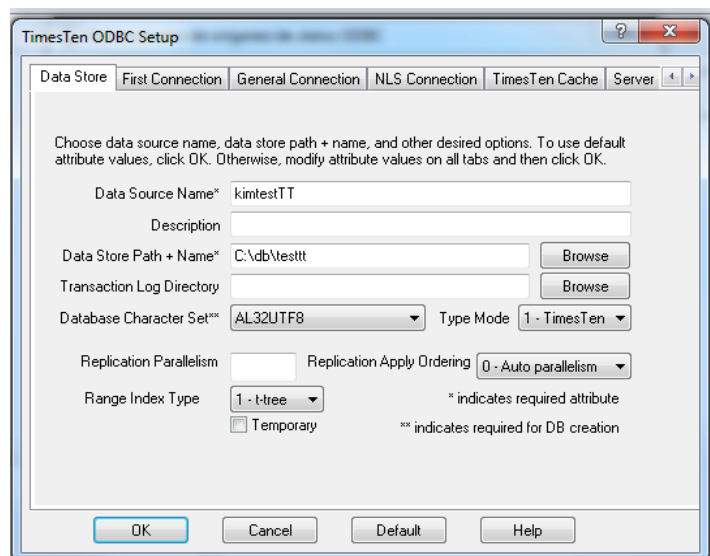
- Search in windows for ODBC and select (Origenes de datos ODBC).



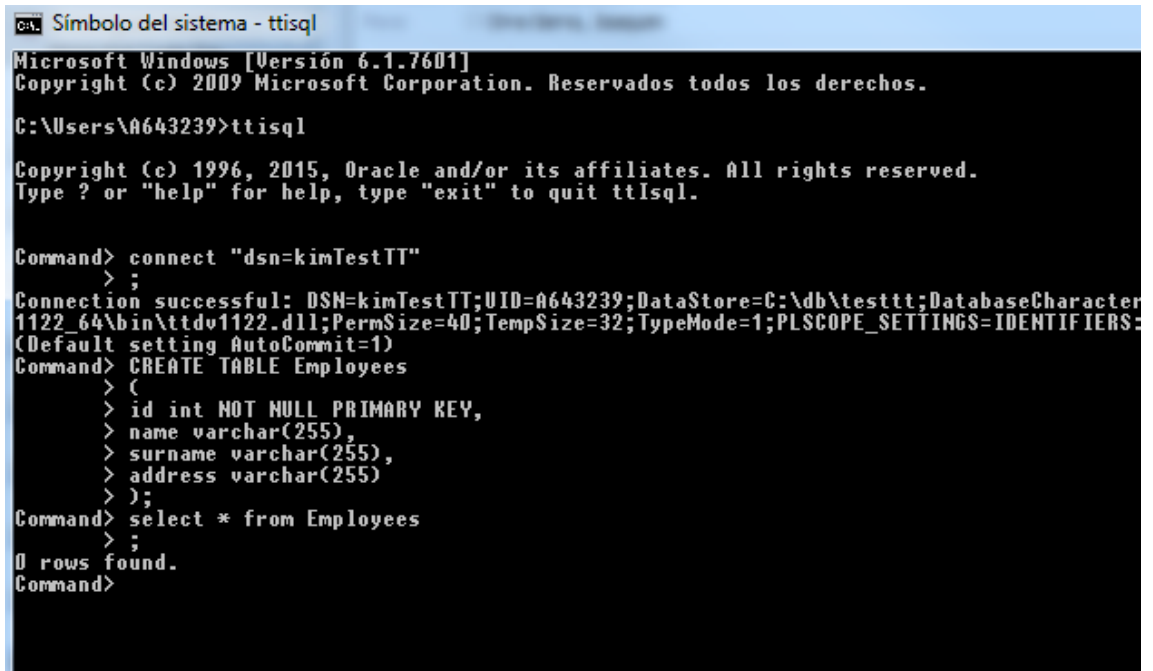
- Select the System DSN tab and click on add (Agregar).



- Select "Times ten Data Manager XX" and click on finish(Finalizar).



- In order to test the connection to the datasource. There is an test in the project called *ConnectionTest* where you can try the connection.
- Once created the DSN open cmd and type:
 - - *ttisql*
 - - *connect "dsn= kimtt";*
 - - Create the table and sequence that we need for the tutorial. The SQL is in *db.sql*, within the project.



```

C:\Users\A643239>ttisql

Microsoft Windows [Versión 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Reservados todos los derechos.

C:\Users\A643239>ttisql

Copyright (c) 1996, 2015, Oracle and/or its affiliates. All rights reserved.
Type ? or "help" for help, type "exit" to quit ttisql.

Command> connect "dsn=kimTestTT"
>
Connection successful: DSN=kimTestTT;UID=A643239;DataStore=C:\db\testtt;DatabaseCharacter
1122_64\bin\tttdv1122.dll;PermSize=40;TempSize=32;TypeMode=1;PLSCOPE_SETTINGS=IDENTIFIERS
(Default setting AutoCommit=1)
Command> CREATE TABLE Employees
> (
>   id int NOT NULL PRIMARY KEY,
>   name varchar(255),
>   surname varchar(255),
>   address varchar(255)
> );
Command> select * from Employees
>
0 rows found.
Command>

```

- Now we should install the driver in maven repository in order to be able to use it later on. Run the following command from the *C:\TimesTen\tt1122_64\lib* folder.

```

mvn install:install-file -DgroupId=com.timesten -DartifactId=ttjdbc7 \
-Dversion=1 -Dpackaging=jar -Dfile=ttjdbc7.jar -DgeneratePom=true

```

Info

* Create the DSN (Official site)

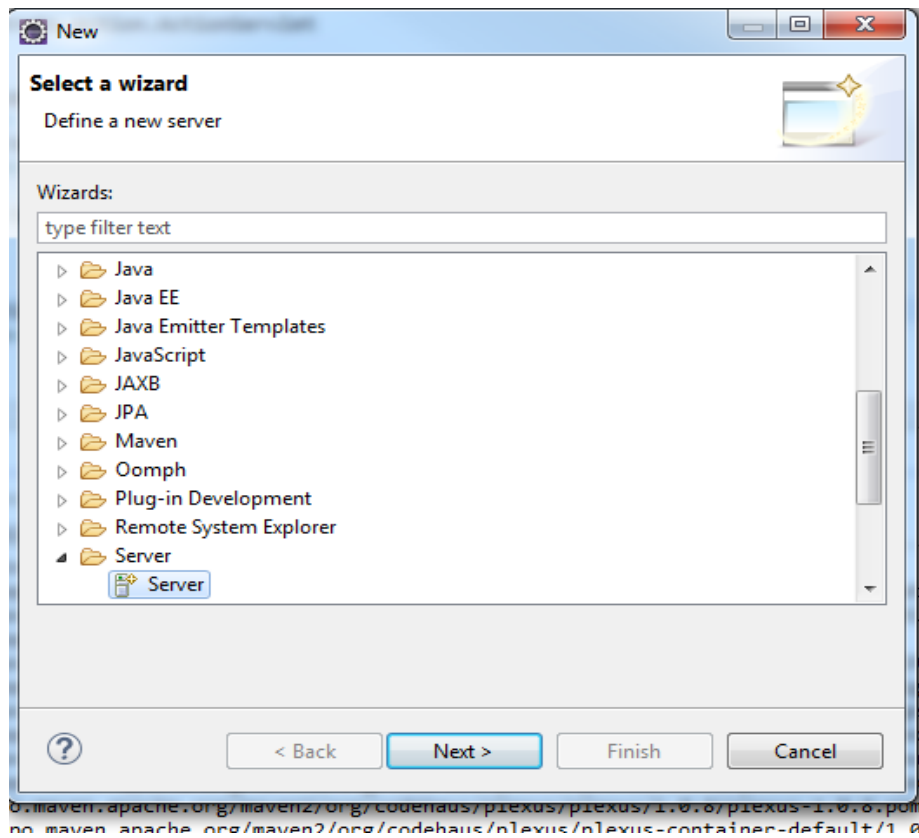
https://docs.oracle.com/cd/E11882_01/timesten.112/e21633/using.htm#TTOPR131

* TimesTen more info in:

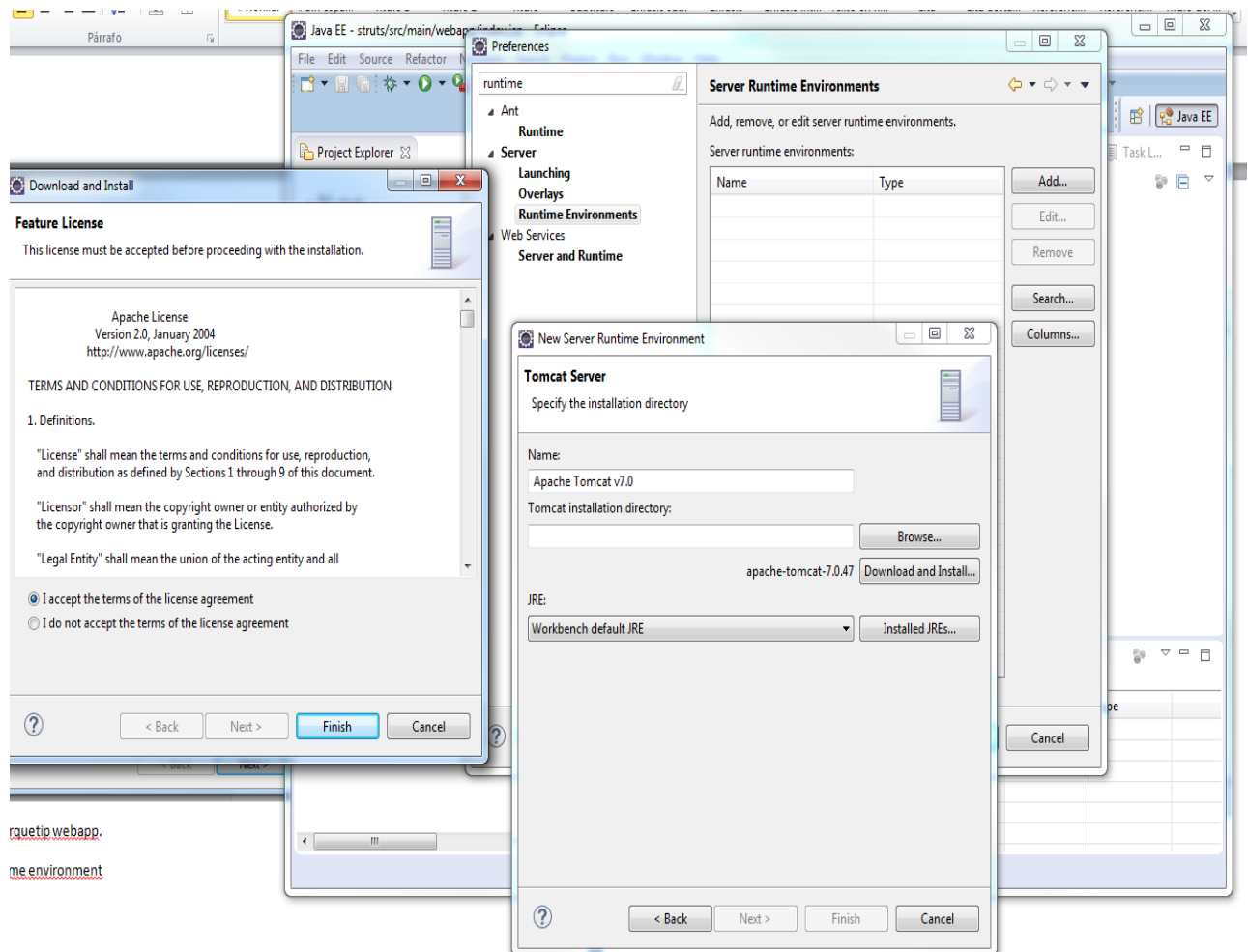
http://download.oracle.com/otn_hosted_doc/timesten/701/TimesTen-Documentation/java_dev.pdf

4. - CREATE TOMCAT SERVER IN ECLIPSE

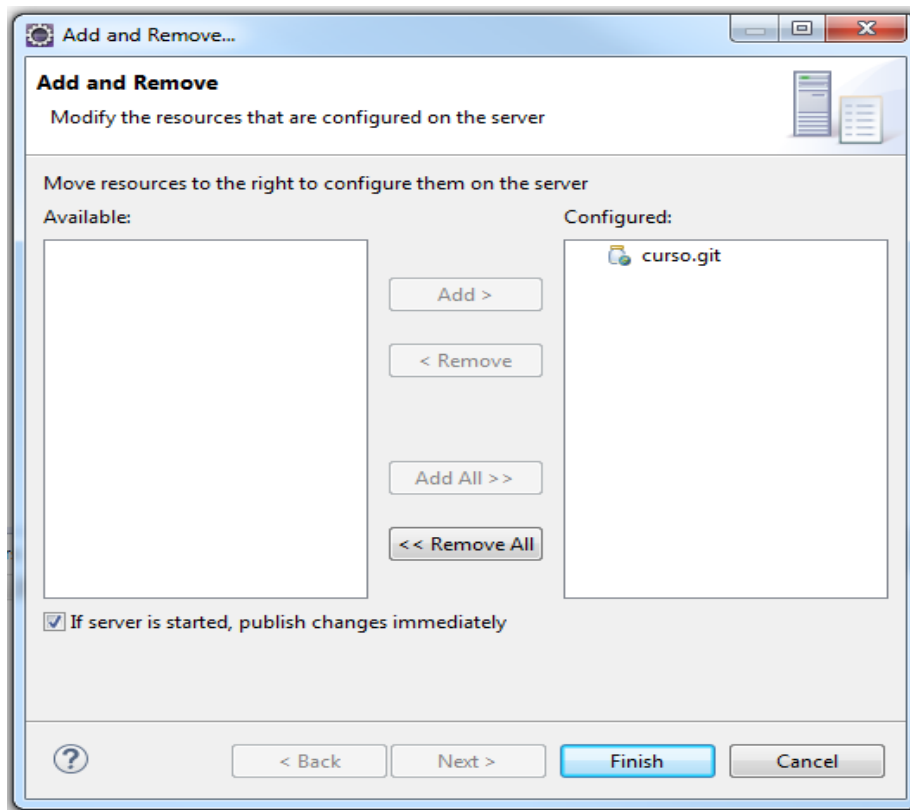
Right click in Servers tab in eclipse and select: new -> Server.



Select Tomcat 7 from the list. (Click auto install if you do not have it installed)



Add the project in the server.

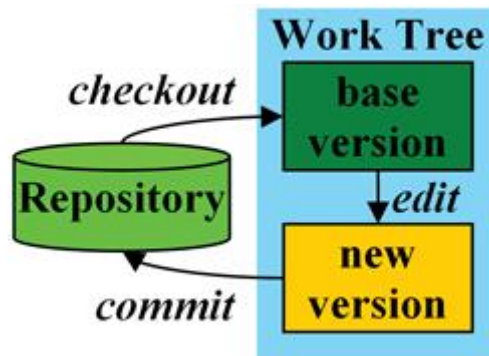


Once is in the server right click on it and click first on publish and then in start/debug and you will see the project running in <http://localhost:8080/struts/welcome.do>

TECHNOLOGY INFORMATION

GIT/SVN/MERCURIAL

All of them are source control systems. Each one has its own approach for the same problem.



The difference between svn (Subversion) and git.

<http://stackoverflow.com/questions/871/why-is-git-better-than-subversion>

Git command line

<https://git-scm.com/docs/gittutorial>

SVN command line

https://tortoisesvn.net/docs/nightly/TortoiseSVN_en/tsvn-cli-main.html

JQUERY/ JQUERYUI

It is a javascript library/framework. It will help us to code our frontend.

¡IMPORTANT TO KNOW!

-Avoid problems by:

- loading first JQuery and then JQueryUI.
- Having only JQuery in the page.

CODING PRACTICES

- As much as we can, do not code inline jquery and javascript. This practice makes page dirty and javascript code not debugable.

```

<%@page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Dispatched 1</title>
</head>
<body>
Disatched 1!
</body>
<script type="text/javascript">
//JQUERY AND JAVASCRIPT
</script>
</html>

```

-Construct your javascript objects like:

```

var yourObject = {
    firstVar: null,
    _init: function(){
        //Constructor
    },
    firstFunction: function() {

    },
};

$(function() {
    yourObject._init();
});

```

*As in javascript there is no limitation while accessing methods within an object. We can mark all private functions and variables in your object with an underscore prefix.

*The “_init” function is the constructor.

*Call the init function on document ready. In order to do a correct binding we need to know that the html has been loaded.

- Do not set javascript functions in the html. It makes code more difficult to maintain. The binding should be done in the javascript object within the init method.

```

<button onclick="myJavaScriptFunction()"></button>
</body>

```

URL for work: <http://localhost:8080/struts/jquery.do>

URL for JQuery info: <https://jquery.com/>

URL for JQuery UI demos (VERY HELPFUL): <http://jqueryui.com/demos/> and http://www.w3schools.com/jquery/jquery_examples.asp

MAVEN

Maven will make life easy by helping us Building the project and importing dependencies.

pom.xml, is the file that will take care of the project. In here we will set up the project build and the dependencies that we need. The POM file is and XML file that contains information about the project such us dependencies, version, packaging, name, etc.

.m2, is the folder within our user that will have all the dependencies that we are requesting to the central repository.

settings.xml, is the file within the .m2 folder that will hold the configuration shared across all the projects.(example a plugin to compile projects that is used in several projects).

Central repository is the cloud from where we will fetch libraries.(Some companies have its own repositories)

Commands, useful commands to run from eclipse or command line:

mvn clean: Cleans the target folder which is the one holding compiled code.

mvn install: Will run testing , compile and create the generated compiled zip (war, jar).

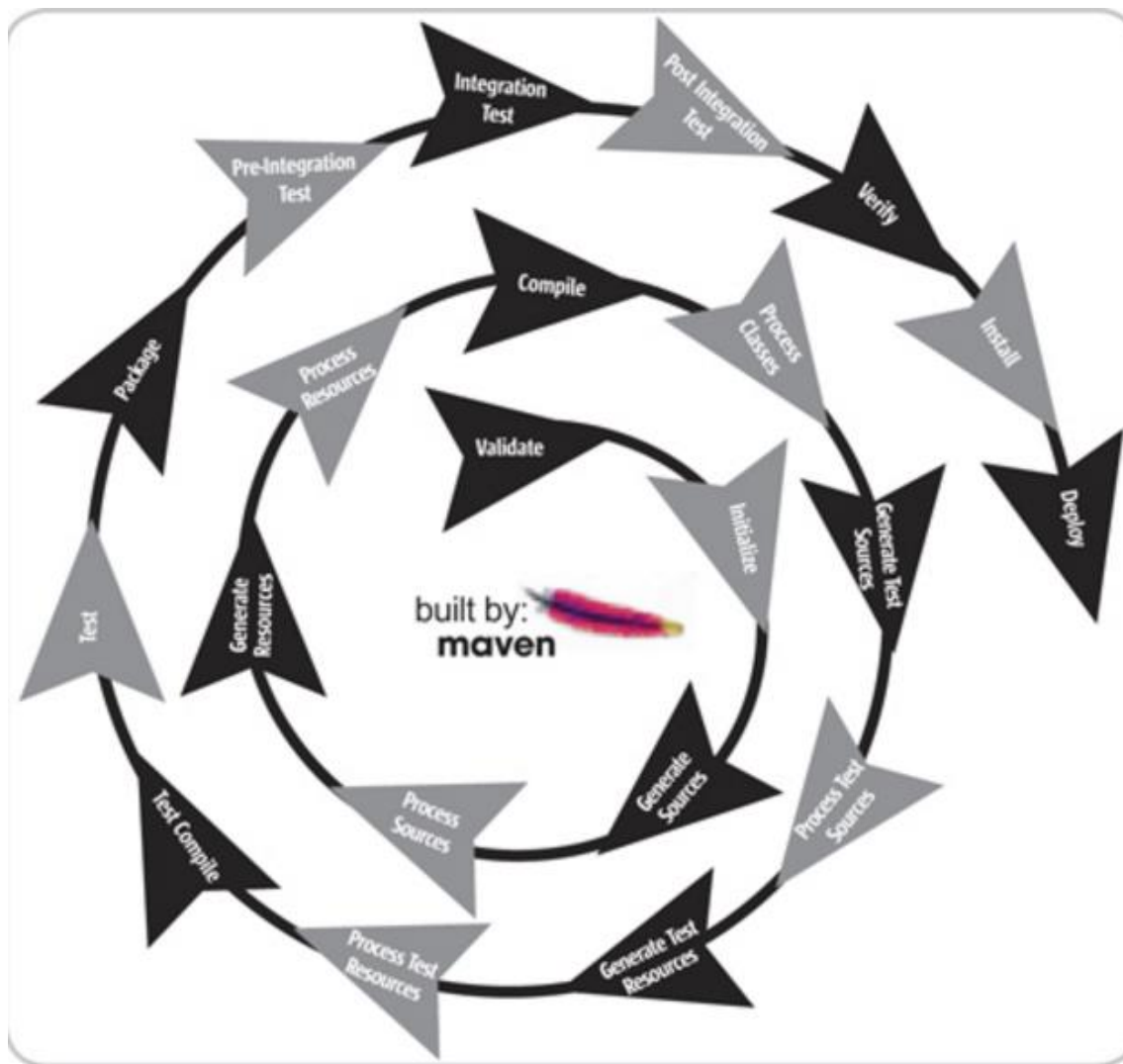
mvn package: Generates the compiled zip (war, jar).

mvn test: Runs the project tests.

PHASES STARTEGY

In maven the goal is the instruction that goes just after the mvn command. If we tell maven to run one goal it will run all the goals that are below this one unless we specifically tell maven not to do it.

Next diagram explains that:



PLUGINS

Maven provides plugins in order to change and add some functionality and behavior while running a goal/phase.

maven-compiler-plugin

This plugin help us while compiling our code.

```
<plugin>
  >> <groupId>org.apache.maven.plugins</groupId>
  >> <artifactId>maven-compiler-plugin</artifactId>
  >> <version>3.5.1</version>
  >> <configuration>
  >>   <!-- or whatever version you use -->
  >>   <source>1.7</source>
  >>   <target>1.7</target>
  >> </configuration>
</plugin>
```

In here for example we are telling which java versions are we using for compiling the code.

maven-surefire-plugin

This plugin will help us while running tests.

```

<plugin>
  >> <groupId>org.apache.maven.plugins</groupId>
  >> <artifactId>maven-surefire-plugin</artifactId>
  >> <version>2.19.1</version>
  >> <configuration>
  >>   <skipTests>>false</skipTests>
  >> </configuration>
</plugin>

```

For example, we are telling here that every time we do “mvn install” we want to run tests as well.

INFO

MAVEN URL: <https://maven.apache.org/guides/getting-started/>

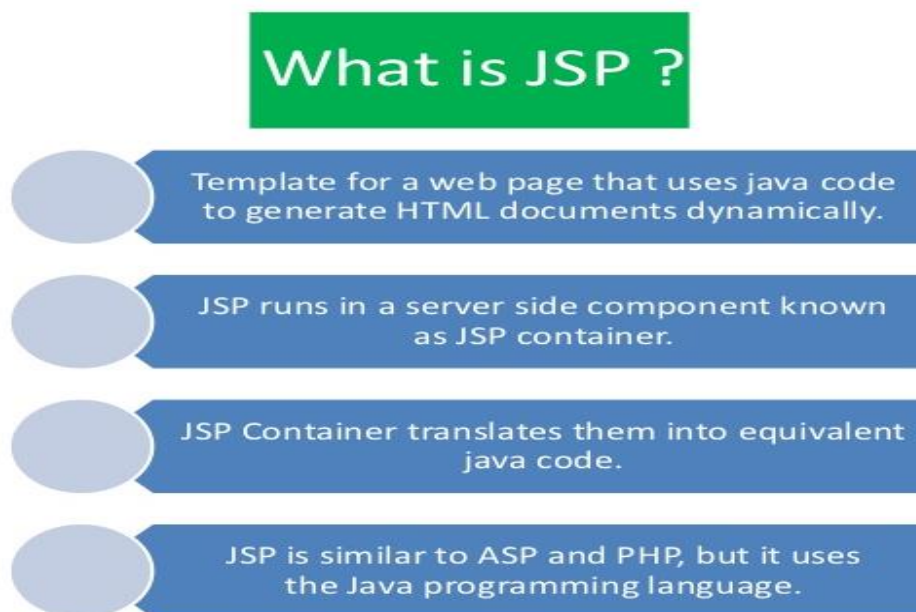
SUREFIRE PLUGIN: <http://maven.apache.org/components/surefire/maven-surefire-plugin/>

MAVEN COMPILER PLUGIN : <https://maven.apache.org/plugins/maven-compiler-plugin/>

JSP

While working with Java we might use JSPs. JSP is basically an HTML with special tags.

- These tags are being run in the server before getting to the client through the network.
- These tags are developed in java.



The main tags we have available in a JSP are very well explained in the following page:

http://www.tutorialspoint.com/jsp/jsp_standard_tag_library.htm

We will see that struts framework has its own tags as well.

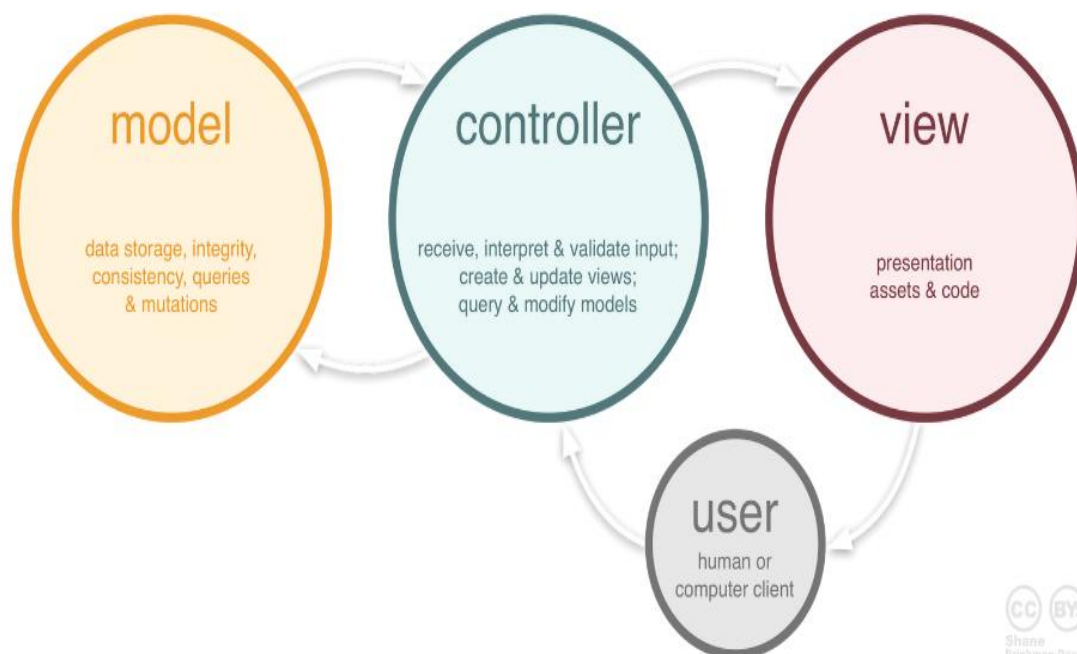
STRUTS

Struts is an open source framework. It is based in the Model, View, Controller (MVC) architecture. It enables you to create maintainable, extensible, and flexible web applications based on standard technologies, such as JSP pages, JavaBeans, resource bundles, and XML.

In order to add struts 1.3 to our project we have to add the following maven libraries:

```
>> <!-- Struts -->
>> <dependency>
>> >> <groupId>org.apache.struts</groupId>
>> >> <artifactId>struts-core</artifactId>
>> >> <version>1.3.10</version>
>> </dependency>
>> <dependency>
>> >> <groupId>org.apache.struts</groupId>
>> >> <artifactId>struts-extras</artifactId>
>> >> <version>1.3.10</version>
>> </dependency>
>> <dependency>
>> >> <groupId>org.apache.struts</groupId>
>> >> <artifactId>struts-taglib</artifactId>
>> >> <version>1.3.10</version>
>> </dependency>
```

MVC



STRUTS PROJECT BASIC CONFIGURATION

In order to configure struts in a project we have to change the web.xml to route certain calls through struts servlet. In the following example all requests matching *.do will be routed

through struts. So that, a call like “/somecall.html” will not go through struts and that can be a simple java servlet.

```
5 <web-app>
6   <display-name>Curso struts</display-name>
7   <servlet>
8     <servlet-name>action</servlet-name>
9     <servlet-class>
10      org.apache.struts.action.ActionServlet
11    </servlet-class>
12    <init-param>
13      <param-name>config</param-name>
14      <param-value>
15        /WEB-INF/struts-config.xml
16      </param-value>
17    </init-param>
18    <load-on-startup>1</load-on-startup>
19  </servlet>
20
21  <servlet-mapping>
22    <servlet-name>action</servlet-name>
23    <url-pattern>*.do</url-pattern>
24  </servlet-mapping>
25 </web-app>
```

The file that takes care of struts configuration is usually called **struts-config.xml**. In this file we basically tell the routing. For instance, the routing the image below will behave as: Any request “/myAction.do” will go through the java class “MyWelcomeAction” and then if success render the jsp “mySuccess.jsp” .

```
<action path="/myAction" type="com.atos.curso.actions.MyWelcomeAction"
        name="myForm" scope="session">
    <forward name="success" path="/mySuccess.jsp" />
</action>
```

GLOBAL EXCEPTIONS

Struts is able to redirect exceptions to specific error pages. So that we can avoid seeing screens like the following one:

localhost:8080/struts/myNullPointerExceptionAction.do

Aplicaciones SarePoint Zen Mis Aplicaciones URA vpn sql

Estado HTTP 500 - java.lang.NullPointerException

type Informe de Excepción

mensaje java.lang.NullPointerException

descripción El servidor encontró un error interno que hizo que no pudiera rellenar este requerimiento.

excepción

```

javax.servlet.ServletException: java.lang.NullPointerException
    org.apache.struts.chain.ComposableRequestProcessor.process(ComposableRequestProcessor.java:286)
    org.apache.struts.action.ActionServlet.process(ActionServlet.java:1913)
    org.apache.struts.action.ActionServlet.doGet(ActionServlet.java:449)
    javax.servlet.http.HttpServlet.service(HttpServlet.java:621)
    javax.servlet.http.HttpServlet.service(HttpServlet.java:728)
    org.apache.tomcat.websocket.server.WsFilter.doFilter(WsFilter.java:51)

```

causa raíz

```

java.lang.NullPointerException
    com.atos.curso.actions.MyNullPointerExceptionAction.execute(MyNullPointerExceptionAction.java:16)
    org.apache.struts.chain.commands.servlet.ExecuteAction.execute(ExecuteAction.java:58)
    org.apache.struts.chain.commands.AbstractExecuteAction.execute(AbstractExecuteAction.java:67)
    org.apache.struts.chain.commands.ActionCommandBase.execute(ActionCommandBase.java:51)
    org.apache.commons.chain.impl.ChainBase.execute(ChainBase.java:191)
    org.apache.commons.chain.generic.LookupCommand.execute(LookupCommand.java:305)
    org.apache.commons.chain.impl.ChainBase.execute(ChainBase.java:191)
    org.apache.struts.chain.ComposableRequestProcessor.process(ComposableRequestProcessor.java:283)
    org.apache.struts.action.ActionServlet.process(ActionServlet.java:1913)
    org.apache.struts.action.ActionServlet.doGet(ActionServlet.java:449)
    javax.servlet.http.HttpServlet.service(HttpServlet.java:621)
    javax.servlet.http.HttpServlet.service(HttpServlet.java:728)
    org.apache.tomcat.websocket.server.WsFilter.doFilter(WsFilter.java:51)

```

nota La traza completa de la causa de este error se encuentra en los archivos de diario de Apache Tomcat/7.0.47.

Apache Tomcat/7.0.47

The exception configuration in struts might look like the following one:

```

</form-beans>

<global-exceptions>
    <exception key="error.global.message" type="java.io.IOException"
        path="/error.jsp" />
    <exception key="error.global.message" type="javax.naming.NoPermissionException"
        handler="com.atos.curso.handlers.PermissionExceptionsHandler"
        path="/error.jsp" />
</global-exceptions>

<action-mappings>

```

It should go between the forms and actions mappings configuration. This configuration will redirect any IOException happening in the project to the *error.jsp* page and any NoPermissionException through the handler and then render the *error.jsp* page. The handler might be useful if we want to do something extra when this exception happens.

ACTION MAPPINGS

The action mappings will tell the route of each request.

Simplest mapping

The simplest mapping configuration is when we do not need a controller. So, a URL maps to a page without any java code between. Used for static pages.

```

<action path="/jquery" type="org.apache.struts.actions.ForwardAction"
    parameter="/jqueryExamples.jsp" />

```

Mapping with an action

If we need a java action to take place between we should change the type attribute in the configuration:

```
<action path="/myAction" type="com.atos.curso.actions.MyWelcomeAction"
        name="myForm" scope="session">

    <forward name="success" path="/mySuccess.jsp" />
</action>
```

*In this example a URL ended with myAction.do will go through the execute method within "MyWelcomeAction" java class and the will render the page "mySuccess.jsp".

Render different pages

Imagine that we want to render a different page depending on some condition in the java Action method.

The struts-config.xml action can be something like:

```
<action path="/someAction" type="com.atos.curso.actions.SomeAction"
        name="employeeForm">

    <forward name="success" path="/mySuccessSave.jsp" />
    <forward name="anotherSuccess" path="/myAnotherSuccessSave.jsp" />
</action>
```

The Java Action method can tell which forward do we want.

```
public ActionForward execute(ActionMapping mapping, ActionForm form, HttpServletRequest request,
    HttpServletResponse response) throws Exception {

    if(someCondition){
        return mapping.findForward("anotherSuccess");
    }else {
        return mapping.findForward("success");
    }

}
```

Using forms with java validation

When we work with forms we have to tell that to struts. Imagine that we have a form in a page to save employees and the post save action is "/saveEmployee.do". We want to validate the form and we want to render a success page once the employee has been saved.

The struts-config.xml should have:

The form defined.

```

<form-beans>
    <form-bean name="employeeForm" type="com.atos.curso.forms.EmployeeForm" />
</form-beans>

```

The struts action mapping with the form reference name.

```

<action path="/saveEmployee" type="com.atos.curso.actions.MySaveAction"
        name="employeeForm" validate="true" scope="request" input="/mySuccess.jsp">

    <forward name="success" path="/mySuccessSave.jsp" />
</action>

```

***validate** attribute tells struts we want to validate the form before getting to the action.

***scope** can be request or session. We set that to request because the form must be available only in the very next request.

***input** If there are errors in the form struts will render again the input page.

The java form class is a normal POJO extending “*FormAction*” Struts internal class and overriding the validation and reset methods.

```

public class EmployeeForm extends ActionForm {

    private String name;
    private String surname;
    private String address;

    @Override
    public ActionErrors validate(ActionMapping mapping, HttpServletRequest request) {

        ActionErrors errors = new ActionErrors();

        if (getName() == null || ("".equals(getName())) {
            errors.add("common.name.err", new ActionMessage("error.common.name.required"));
        }

        return errors;
    }

    @Override
    public void reset(ActionMapping mapping, HttpServletRequest request) {
        name = "";
        surname = "";
        address = "";
    }
}

```

The form should look like:

```

<html:form action="/saveEmployee.do">

    <div class="error" >
        <html:errors />
    </div>
    <html:text property="name" size="20" maxlength="20" />
    <html:text property="surname" size="20" maxlength="20" />
    <html:text property="address" size="20" maxlength="20" />

    <div style="padding: 16px">
        <div style="float: left; padding-right: 8px;">
            <html:submit>
                Save Employee
            </html:submit>
        </div>
        <html:reset>
            Reset form
        </html:reset>
    </div>
</html:form>

```

*The tag html is a specific struts tag.

Using forms with XML validation

Imagine that we want something like number 4 example but we want to use an xml file for validation instead of a java code. We can do that by using a struts validation plugin.

The struts-config.xml should have:

The form defined.

```

» <form-bean name="employeeXMLValidationForm"
» > > type="com.atos.cursor.forms.EmployeeXMLValidationForm" />
</form-beans>

```

The action mapping which is the same as any other one.

```

<action path="/saveWithXMLValidation" type="com.atos.cursor.actions.SaveWithXMLValidationAction"
» name="employeeXMLValidationForm" input="/xmlValidation.jsp">
» <forward name="success" path="/mySuccessSave.jsp" />
</action>

```

The plugin configuration at the very end of the struts-config.xml file, just before closing struts-config xml tag.

```

<plug-in className="org.apache.struts.validator.ValidatorPlugIn">
» <set-property property="pathnames"
» > > value="/WEB-INF/validator-rules.xml, /WEB-INF/validator-employee.xml" />
</plug-in>

```

In here we have to tell struts which are the:

- Validation rules files. This file contains some general rules to be used when validating. We can create our own rules as well.
- Our validator xml files which will have specific rules for specific forms.

The java form class is a normal POJO extending “*ValidatorForm*” Struts internal class instead of “*FormAction*” which is used in normal forms.

With XML Validation

```
public class EmployeeXMLValidationForm extends ValidatorForm {
```

Without XML validation

```
public class EmployeeForm extends ActionForm {
```

Regarding to the html form should look like the one in example 4.

Multiple actions

Multiple actions within the same java class. Holding multiple actions in the same java class is easy. I have to change the struts internal class that we extend in normal Action controllers.

While coding your controller, instead of using *Action* use *DispatchAction*.

```
public class MyDispatchAction extends Action {
```

```
public class MyDispatchAction extends DispatchAction {
```

Your struts-config.xml file action mapping should be something like:

```
<action path="/MyDispatchAction" type="com.atos.curso.actions.MyDispatchAction"
    parameter="action">
    <forward name="page1" path="/dispatched1.jsp" />
    <forward name="page2" path="/dispatched2.jsp" />
</action>
```

*The parameter attribute is the name of the query parameter in the URL specifying the action. For example, “/struts/MyDispatchAction.do?action=page2”.

STRUTS JSP TAGS

Struts has special html tags in order to help us developing. All tags generate html code, they run in the server, not in the client!!.

The java library that holds these classes can be downloaded from MAVEN repository:

```
» <dependency>
»   <groupId>org.apache.struts</groupId>
»   <artifactId>struts-taglib</artifactId>
»   <version>1.3.10</version>
» </dependency>
```

URL http://tutorials4u.net/struts-tutorial/struts_logic_tag_example.html

STRUTS INTEGRATION TESTING

In order to be able develop out integration tests in Struts we should, at least, add the following maven libraries:

```
» <!-- Testing -->
» <dependency>
»   <groupId>junit</groupId>
»   <artifactId>junit</artifactId>
»   <version>4.11</version>
»   <scope>test</scope>
» </dependency>
» <dependency>
»   <groupId>strutstestcase</groupId>
»   <artifactId>strutstestcase</artifactId>
»   <version>2.1.4-1.2-2.4</version>
»   <scope>test</scope>
» </dependency>
» <dependency>
»   <groupId>org.apache.struts</groupId>
»   <artifactId>struts-tiles</artifactId>
»   <version>1.3.10</version>
»   <scope>test</scope>
» </dependency>
» <dependency>
»   <groupId>commons-collections</groupId>
»   <artifactId>commons-collections</artifactId>
»   <version>3.2</version>
»   <scope>test</scope>
» </dependency>
```

- Note the scope. Setting the scope to test means that these libraries are only available when testing.

The following class is an example of an integration test:

```

public class MySaveActionTest extends MockStrutsTestCase {

    » public MySaveActionTest(String testName) {
    »     » super(testName);
    » }

    » public void setUp() throws Exception {
    »     » super.setUp();
    »     » setConfigFile("/WEB-INF/struts-config.xml");
    » }

    » protected void tearDown() throws Exception {
    »     » super.tearDown();
    » }

    » public void testSaveAction_GoesBackToFormPageWithErrorMessage() {
    »     » setRequestPathInfo("/saveEmployee.do");
    »     » actionPerform();
    »     » verifyForwardPath("/mySuccess.jsp");
    »     » verifyActionErrors(new String[] { "error.common.name.required" });
    » }

    » public void testSaveAction_CreatesEmployee() {
    »     » addRequestParameter("name", "vikas");
    »     » addRequestParameter("surname", "vikas surname");
    »     » addRequestParameter("address", "Vikas address");
    »     » setRequestPathInfo("/saveEmployee.do");
    »     » actionPerform();
    »     » verifyForwardPath("/mySuccessSave.jsp");
    »     » verifyNoActionErrors();
    » }

}

```

- All our **Struts integration tests** must extend the *MockStrutsTestCase* java class.
- In the **setup method** we should specify some global configuration for struts or some initialization that needs to be run before testing.
 - The method **setConfigFile** specifies the path to the struts configuration file. Notice that the route in this test is the default one, so that, we could just delete the line and the test would run anyway.
- The test ...***GoesBackToFormPageWithErrorMessage*** test the action when no form has been set.
- The test ...***CreatesEmployee*** test the action with a form added to the request.

However, this test has a big problem!!. It is not encapsulated. It depends on an existing DDBB connection. For building an encapsulated integrated test we should be mocking the *TtConnection* class. Sadly this class is a singleton...

<http://stackoverflow.com/questions/2302179/mocking-a-singleton-class>

INFO

In the following page you will find some useful struts examples:

<http://www.mkyong.com/struts/struts-hello-world-example/>

CLEAN CODE

Bad code does too much – Clean code is focused

Reading your code should be pleasant

Can be easily extended by any other developer

It should have minimal dependencies

Smaller is better

It should have unit and acceptance tests

Expresses the intent of the programmers